Blem.

C:\Users\Ray\Desktop\laManoMalvadita\The Dinosaur Game\TrashcanLogo.tif

The Trashcan

**Software and Media Publication**

Computer Game Design Document and Development Log

|  |
| --- |
| The Dinosaur Game Development Team  9 February 2018 to 20 July 2018, 12 February 2019 to Future Date |

The Dinosaur Game

|  |
| --- |
| The Trashcan (Software and Media Publication, NFP) |
| The Dinosaur Game |
| Computer Game Design Document and Development Log |

|  |
| --- |
| The Dinosaur Game Development Team  9 February 2018 to 20 July 2018, 12 February 2019 to Future Date |

All text herein, software described, music, and visual artwork Copyright © 2018, 2019 Raymond Arias, Jr. and Maria A. Arias (and where applicable, Stanton J. Viaduc).

Although a number of nonproprietary licenses are being considered for the release of the final work, none of these licenses apply at this stage.

Neither the authors, nor their agencies nor nonprofit corporations, as connected to this document, such as Arias Concepts & Services or The Trashcan Software and Media Publication, are liable for any unauthorized and/or improper use of these materials.

Guide to Terminogy Specific to This Document

**Development Team**

**Name in Document** **Full Name** **Roles in Development**

Ray Raymond Arias, Jr. Developer of Original Version, Primary

Developer of New Version, Primary Author

of this game document, Direction, Artwork, Coding

marias María A. Arias Development Consultant, Artwork, Visual

Consultant, Other Roles To Be Determined

Stan Stanton J. Viaduc Development Consultant, Other Roles To Be

Determined

**Planned Versions and Supplements**

**Name in Document** **Full Name** **Description**

TOG The Dinosaur Game: The Original Game Depending on the context: 1. The original 1986 version of the game coded in Microsoft Commodore BASIC 7.0 and Motorola 6502 machine code by Ray, or, 2. The preliminary version of the prototype for The Dinosaur Game that has been in development for this project, being coded in C++ for Windows PC by Ray. (This preliminary version will reproduce most of the functionality of the 1986 version of this game.) (This is the current stage of development.) To eliminate any source of confusion, 1. Is also called “the (original) 1986 version,” “the (original) 1986 game,” or “TOG 1986,” and 2. is also called “the preliminary version,” “the initial prototype,” or “TOG Prototype.”

TNG The Dinosaur Game: The New Game The version that will be made for release as soon as development of the preliminary version of TOG has concluded. (This stage of development has not been reached yet.)

TerrestrialGravitation TerrestrialGravitation, a C++ Library A software library that assists in making calculations to plot graphical object on screen and makes them appear they are reacting to terrestrial gravity. (TerrestrialGravitation is being developed by Ray at the same time as TOG and should be ready for use when work on TNG begins.)

Supplemental Paper Applying Mathematics and Gravitational Terrestrial Motion in 2D Settings: a Physics Supplement for Applications to Software, Including a Primer or Review of Analytical Geometry, Basic Calculus, and Newtonian Kinematics

An instructional paper introducing the mathematical and physical concepts necessary to understand motion and terrestrial gravity in a two-dimensional setting with an appendix explaining the classes and functions in the TerrestrialGravitation library.

Table of Contents

4O. Guide to Terminology Specific to This Document……...……………………..…….ix

1. Development Team............................................................................................ix
2. Planned Versions and Supplements...................................................................ix

OOO. Table of Contents………...………………………………....…...………...………......xi

OO. Table of Illustrations………………………………………………………………...…xiii

[O. Abstract 1](#_Toc1320822)

[A. Description and Introduction 1](#_Toc1320823)

[B. History 1](#_Toc1320824)

[C. Planning for Development of Prototype 1](#_Toc1320825)

[D. Conceptual Development of Newer Modernized Game 2](#_Toc1320826)

[E. Development of the Initial Prototype 2](#_Toc1320827)

[I. History 2](#_Toc1320828)

[A. Original 1986 Game Look 2](#_Toc1320829)

[B. Original Game Music 3](#_Toc1320830)

[C. Original Game Play 5](#_Toc1320831)

[1. Description in Prose 5](#_Toc1320832)

[2. Pseudocode 7](#_Toc1320833)

[D. Original Game Story 9](#_Toc1320834)

[II. Planning for Development of Prototype 10](#_Toc1320835)

[A. Basic Plan for Proto type 11](#_Toc1320836)

[B. Filling in Pseudocode to Recreate TOG 11](#_Toc1320837)

[C. Supplemental Paper on Gravitation 11](#_Toc1320838)

[III. Conceptual Development of Newer Modernized Game 12](#_Toc1320839)

[A. Discussion of Possible Change to Game Title 12](#_Toc1320840)

[B. Ideas to Fill in the Story 12](#_Toc1320841)

[1. First Stage/ Volume One 13](#_Toc1320842)

[2. Second Stage/Volume Two 14](#_Toc1320843)

[3. Third Stage/Volume Three 14](#_Toc1320844)

[IV. Development of the Initial Prototype 15](#_Toc1320845)

[A. Initial Stages 15](#_Toc1320846)

[B. Use of Simple DirectMedia Layer (SDL) and Abandonment of a Class 16](#_Toc1320847)

[C. Sprite Image Assets 17](#_Toc1320848)

[Appendix 1: Letters for Designation of Stage of Software Development Cycle 19](#_Toc1320849)

[Appendix 2: Development Log 20](#_Toc1320850)

**Table of Illustrations**

[Figure 1: Initial screen of original 1986 version of The Dinosaur Game, recreated from memory by Ray using Windows Paint. 3](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320285)

[Figure 2: Sketch of Giganotosaurus digitally drawn with Autodesk Sketchbook and GIMP (GNU Image Manipulation Program) by marias. 5](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320286)

[Figure 3: Recreation of game play screen from TOG with the path of the spear shown, digitally drawn using Windows Paint from memory by Ray. 6](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320287)

[Figure 4: Rough sketch for the game, digitally drawn using Iskn Slate2 by Ray. 10](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320288)

[Figure 5: Preliminary sketch for the game, digitally drawn using Autodesk Sketchbook by marias. 13](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320289)

[Figure 6: Screen capture showing the result of Ray's first successful compilation of C++ code using the program library SDL 2.0.8. 16](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320290)

[Figure 7: A sample BMP images of the player, the dinosaur, and the spear, all digitally drawn by Ray using Windows Paint. 18](file:///C:\Users\Ray\Desktop\The%20Trashcan%20Software%20&%20Media\Software%20Development\The%20Dinosaur%20Game\Current%20Docs\The%20Dinosaur%20Game-19.docx#_Toc1320291)

O. Abstract

1. Description and Introduction

This document explains and records the development of a computer game whose initial prototype is a reproduction of a game made by Ray in 1986 called The Dinosaur Game. This 1986 game and its newer reproduction are referred to as TOG for “The Original Game.” After the prototype TOG is developed, improvements will be made upon it in order to facilitate development of the next stage, called TNG for “The New Game” (insert jokes referring to *Star Trek: The Next Generation* here). This document exposits most of the ideas, history, rationales, procedures, logic, in-game story, and each stage of development in detail. The purpose of this document is to give a record and explanation of the development of this game for recall and communication for and by The Dinosaur Game Development Team as well as for and by any investors and backers. This document also serves educational purposes for members of the public who may be interested in game development in general or the development of this game specifically.

1. History

The first section tells the story of the original 1986 game, how it looked, the music in it (with sheet music and Lilypond code included), a detailed verbal description, a breakdown of the logic of the 1986 game in pseudocode, and a detailed explanation of the original back-story of the 1986 version of the game. Since no written, printed, nor digital records exist of this game, all this information is provided by the best recollection of Ray, the sole developer of TOG 1986, and the main developer of all the newer versions of The Dinosaur Game.

1. Planning for Development of Prototype

The second section relates the process of how the new initial version of this game, TOG Prototype, is being developed, including the use the SDL (Simple DirectMedia Layer) 2.0.8 software library (this is downloaded outside program code that, when paired up with our program code, simplifies the display of graphics and sound), the possible later use of the Unity game engine (an outside application program that facilitates both game design and game play), how the pseudocode from the previous section is being used to code the prototype, and the introduction of an in-house software library, TerrestrialGravitation, and its accompanying supplemental instructional paper that is being simultaneously developed to facilitate placement and motion of all graphical objects in the game.

1. Conceptual Development of Newer Modernized Game

The fourth section discusses the creation of a new expanded back-story for use in the development in the TNG stage of development. Also, there is a short discussion on possibly changing the name of this game to something more descriptive.

1. Development of the Initial Prototype

The fifth section details how TOG Prototype is being coded in C++, including an explanation of how program functions are being coded to be compatible with SDL 2.0.8 and how SDL is used to draw and place the window and the background graphics. Also, there is discussion on how Ray was most of the way to developing a program class for dealing with how to display text on the screen, but then found a way to do it more efficiently by using the SDL TTF 2 library, and therefore the development of the class was abandoned. Further, there is discussion on the creation and manipulation of the graphical images of the characters in the game, also technically known as sprites.

1. History
2. Original 1986 Game Look

The current design of this game is based on a simple game that Ray made in 1985-6 on a Commodore 128 in Microsoft Commodore BASIC 7.0 and 8502 machine code using the built in machine language monitor. There are absolutely no soft- nor hardcopies of any code, screens, or design records remaining in existence for this original game. The only source for information about this game is Ray’s memory of it. Any reference to this game in this document will be labeled TOG for “The Original Game.”

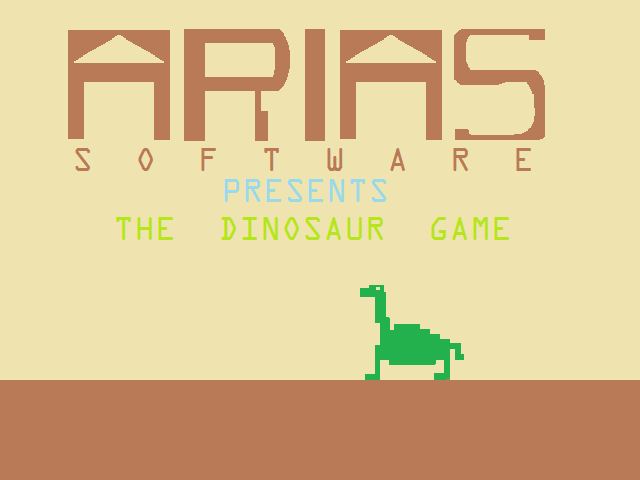
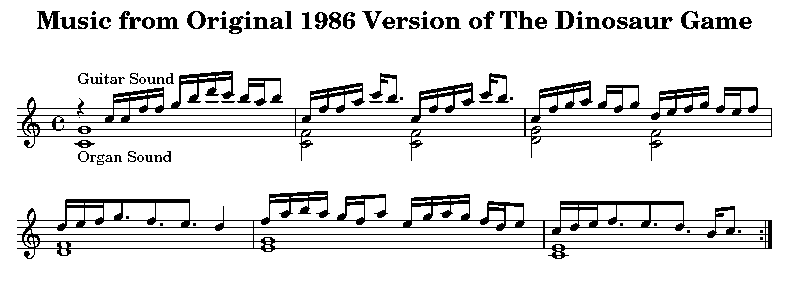
TOG had a start screen with a dark brown ground comprising about the bottom fifth of the screen, a yellow background making up the remainder of the screen and a green dinosaur with a Brontosaurus/Apatosaurus appearance walking across atop the ground from right to left with deep percussive tones played with every step. The size of this dinosaur was approximately a fifth of the screen vertically and a seventh of the screen horizontally, and it was physically about as tall from head to foot as it was long from head to tail. Displayed at about mid-height of the start screen were large letters saying “ARIAS” and under these were medium letters spaced to fit the horizontal space taken up by the large letters saying “S O F T W A R E” and under this were more medium letter saying “Presents THE DINOSAUR GAME.”

Figure 1: Initial screen of original 1986 version of The Dinosaur Game, recreated from memory by Ray using Windows Paint.

1. Original Game Music

The music for TOG was played with the three-voice VIC sound chip in two of its three voices. The lower voice was an organ and the higher voice was a guitar. This is a simple six-bar repeated tune in the key of C, with a chored progression of C, F, (F,) G, F, C, Dm, Em, C. Originally, the music was encoded using PLAY statements in MS Commodore BASIC 7.0 for the C-128. This particular music, however, was encoded using GNU LilyPond 2.18.2.1.



The LilyPond code for this music follows:

%Recreation of Music from the Original 1986 Version of The  
 %Dinosaur Game by Ray Arias

%Coded for GNU LilyPond 2.18.2.1 by Ray Arias, 14-16 April 2018

\version "2.18.2.1"

\header{

title = "Music from Original 1986 Version of The Dinosaur  
 Game" }

\paper {

markup-system-spacing #'basic-distance = #16

system-system-spacing #'basic-distance = #16 }

\score {

\relative c' {

\time 4/4 \repeat volta 8 {

<< \stemDown <c g'>1\_"Organ Sound" \\

\stemUp { f'4\rest ^"Guitar Sound" c16 c f f g b  
 d c b a b8 } >> |

<< \stemDown<c,, f>2\\ \stemUp {c'16 f f a c b8.} >>

<<\stemDown<c,, f>2\\ \stemUp{c'16 ffac b8.} >>|

<< \stemDown <d,, g>2\\ \stemUp {c'16 fgagf g8}>>

<<\stemDown <c,, f>2\\ \stemUp{d'16 efgfe f8}>>|

<< \stemDown<d, f>1\\ \stemUp {d'16 ef g8. fe d4}>>|

<< \stemDown <e, g>1 \\  
 \stemUp { f'16 a b a g f a8 e16 g a g f d e8}>>|

<<\stemDown<c, e>1\\ \stemUp {c'16def8.edb16c8.}>>|}

}

\layout {

indent = 0 \cm

short-indent = 0 \cm

\context {

\Score \remove "Bar\_number\_engraver" }

}

\midi {

\tempo 4 = 75 }

}



Figure 2: Sketch of Giganotosaurus digitally drawn with Autodesk Sketchbook and GIMP (GNU Image Manipulation Program) by marias.

1. Original Game Play
2. Description in Prose

TOG began when the fire button on the joystick was pressed and displayed a dark blue screen with a brown ground bottom in the same position it was in the start screen. The player was displayed on the left side of the screen as a white human-looking figure with what appeared to be a spear in its hand. In the bottom, lettering was displayed as a message line by line to the player explaining the game story (see next subsection, D. Original Game Story), until either the fire button was pressed again or 10 seconds passed after the final line of text. Then a long tone ascending in frequency would be played and, when the tone rose to top pitch, the same yellow background with a brown bottom as the start screen would be displayed, and another descending tone starting at high frequency would be played. When the tone lowered to its bottom pitch, the dinosaur would appear all the way on the right edge of the screen and play would begin at 3 lives and first level.

When the joystick was left alone, the player figure would stay still and the dinosaur figure would proceed left toward the player at a certain speed. (At Level 1, this speed was about 5 pixels a second, but this speed would increase at higher levels.)

When the joystick was pushed left, the player figure went about 20 pixels to the left, up to the left edge of the screen.

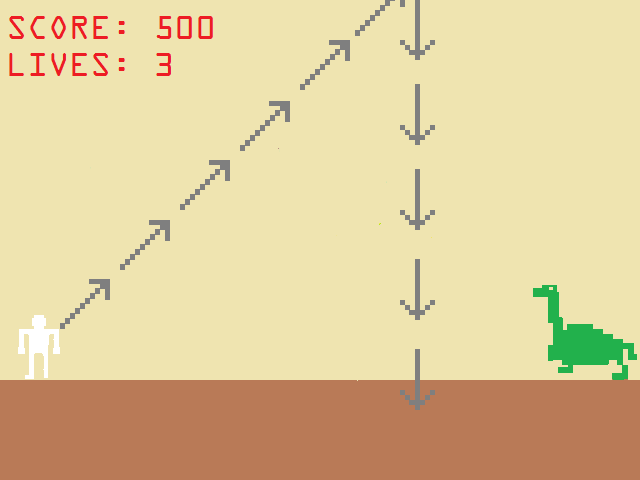
When the joystick was pushed right, the player figure went about 20 pixels to the right, theoretically up to the right edge of the screen, although because the dinosaur was inevitably in the way, this typically was not possible. (Sometimes, a glitch early on in the development of TOG would actually make this possible because the routine that detected whether the player and dinosaur collided would not run fast enough—you gotta love Commodores running BASIC—before the player passed to the other side of the dinosaur. However, this issue was later solved by matching the X position of both the player and the dinosaur in such a way that if the X position of the player was greater than or equal to the X position of the dinosaur, the routine that causes the player to die would execute. However, this method of detection was used concurrently with the traditional collision detection method, as opposed to in place of it.)

Figure 3: Recreation of game play screen from TOG with the path of the spear shown, digitally drawn using Windows Paint from memory by Ray.

Pushing the joystick up or down, while early on in development of TOG had no effect, later had the effect of allowing the dinosaur to proceed somewhat faster during the time the joystick was pushed either up or down. When the joystick was released from the up or down position, the dinosaur would again proceed at its original speed.

When the fire button was pressed, a spear coming from the player’s position would be released and travel at 45° counterclockwise from the horizontal until it got to the top of the screen, at which point the spear would appear to turn 135° clockwise, such that it proceeded straight downward. If the spear did not hit the dinosaur, play would continue as before. However, if the spear did hot the dinosaur, the dinosaur would vanish, the player would be advanced to the right until the player figure wrapped around to the left side again (in what was presumably another scene, but appeared to be the same) and would stop at the same position it was at the beginning of the first level, another dinosaur would appear at the right edge of the screen, and play would start again at a new level.

1. Pseudocode

void DisplayStartScreen() {

DisplayTitleAndBackground;

PlayThemeMusic;

int XPositionDinosaur = RightEdge;

int DinosaurFigure = 1; /\* There are 4 dinosaur figures according to what position the Dinosaur’s feet are in \*/

while (FireButtonNotPressed) {

while (DinosaurFigure <= 4) {

DisplayDinosaur(DinosaurFigure, XPositionDinosaur);

DinosaurFigure++;

XPositionDinosaur -= 5;

if ((FireButtonNotPressed) &&

(XPositionDinosaur <= LeftEdge))

XPositionDinosaur = RightEdge; }

Dinosaurfigure = 1; }

StopThemeMusic;

return; }

void GamePlay() {

DisplayStoryBackground;

While (FireButtonNotPressed)

for (int StoryLine = 0; StoryLine <= LastLineOfStory; StoryLine++) DisplayStoryLine(StoryLine);

DisplayStandardGameBackground;

int Level = 1;

int Score = 0;

int Lives = 3;

int XPositionPlayer = LeftEdge;

int XPositionDinosaur = RightEdge;

int DinosaurFigure = 1;

int PlayerFigure = 2; /\* There are 3 player figures according to if the player is standing still (PlayerFigure = 2), moving to to the left (PlayerFigure = 1), or moving to the right (PlayerFigure = 3). \*/

bool PlayerAlive = TRUE;

bool DinosaurAlive = TRUE;

DisplayPlayer(XPositionPlayer, PlayerFigure);

while ((Level < MaxLevel) && (Lives > 0)) {

while ((DinosaurAlive) && (PlayerAlive)) {

if (NoJoystickMovement) || (JoystickUp) || (JoystickDown)) {

PlayerAlive = (XPositionPlayer < (XPositionDinosaur - MarginOfError));

if (DinosaurFigure <= 4) {

DisplayDinosaur(XPositionDinosaur, DinosaurFigure);

DinosaurFigure++;

XPositionDinosaur -= 5;

if (NoJoystickMovement) Wait(Level);

else if (Level < (MaxLevel – 2))

Wait(Level + 2);

else Wait(MaxLevel); }

else DinosaurFigure = 1; }

else if ((JoystickLeft) || (JoystickDiagUpLeft) || (JoystickDiagDownLeft)) {

PlayerAlive = (XPositionPlayer < (XPositionDinosaur - MarginOfError));

int InitPlayerPosition = XPlayerPosition;

int FinalPlayerPosition = XPlayerPosition – 5;

if (FinalPlayerPosition < LeftEdge)

FinalPlayerPosition = LeftEdge;

PlayerFigure = 1; /\* Facing left \*/

while (XPositionPlayer > FinalPlayerPosition) {

XPositionPlayer--;

DisplayPlayer(PlayerFigure, XPlayPOsition);

}

}

if ((JoystickRight) || (JoystickDiagUpRight) || (JoystickDiagDownRight)) {

PlayerAlive = (XPositionPlayer < (XPositionDinosaur - MarginOfError));

int InitPlayerPosition = XPlayerPosition;

int FinalPlayerPosition = XPlayerPosition + 5;

PlayerFigure = 3; /\* Facing right \*/

while (XPositionPlayer < FinalPlayerPosition) {

XPositionPlayer++;

DisplayPlayer(PlayerFigure, XPlayPOsition);

}

}

else if (FireButtonPressed) {

int YPositionSpear = Ground;

int XPositionSpear = XPositionPlayer;

bool DinoHit = FALSE;

bool SpearFigure = FALSE; /\* There are 2 spear figures, one for shooting up at 45 deg angle and one for going straight down \*/

for (int YPositionSpear = Ground; YPositionSprear > TopOfScreen; YPositionSpear--) {

XPositionSpear++;

DisplaySpear(SpearFigure, XPositionSpear, YPositionSpear); }

SpearFigure = TRUE; /\* Switch to falling spear \*/

For (int YPositionSpear = TopOfScreen; YPositionSpear < Ground; YPositionSpear++)

DisplaySpear(SpearFigure, XPositionSpear, YPositionSpear);

DinoAlive = !(Abs(XPositionSpear – XpositionDinosaur) <= 7); }

}

if (PlayerAlive) { /\* Dinosaur dead \*/

KillDinosaur;

Score += 100;

Level++;

GoNewLevel; }

else { /\* Player dead \*/

KillPlayer;

Lives--; }

}

if (Level == MaxLevel) ShowEndGameGraphics;

return; }

main() {

while(1) {

DisplayStartScreen();

GamePlay(); }

}

1. Original Game Story

TOG’s story was displayed just before game play began. It was written in the second person (speaking as an unidentified narrator referring to the player as “you” giving exposition directly) and was displayed approximately as follows:

YOU ARE A SPECIAL AGENT FOR THE SUPER-SECRET CENTRAL INTELLIGENCE AGENCY (SSCIA). EXTREMELY INTELLIGENT DINOSAURS FROM MILLIONS OF YEARS AGO HAVE INVENTED TIME MACHINES AND ARE USING THEM TO TRAVEL TO THE PRESENT DAY IN ORDER TO KILL OFF ALL HUMANITY AND TAKE OVER THE MODERN WORLD IN ORDER TO ACQUIRE ALL OUR TECHNOLOGY AND INFRASTRUCTURE. THE SSCIA IS SENDING YOU BACK INTO THE DINOSAURS’ ORIGINAL TIME TO ANIHILATE THESE DINOSAURS BEFORE THEY TRAVEL FORWARD IN TIME. YOU MUST USE YOUR ANNIHILATION SPEARS TO TELEPORT THEM INTO OBLIVION BEFORE THEY DEVOUR YOU. THE AGENCY HAS ACQUIRED ONE OF THE DINOSAURS’ TIME MACHINES FOR THIS PURPOSE. PREPARE FOR TRANSPORT.

The basic notion behind the original story is that many dinosaurs were way more intelligent, civilized, and organized than we gave them credit for. Apparently, some of them have developed a method to travel in time and have used it to carefully and inconspicuously travel into the future until they happened upon our time and became envious of all our technology that they desire in order to add to their own. The role of the player is to defeat the dinosaurs before they are able to make this happen. Certain points of background are not explained in the game, such as if the dinosaurs already have their own technology that is advanced enough to travel in time, why would they still desire human technology, and how exactly was the SSCIA able to acquire one of the dinosaurs’ time machines and somehow adapt it for human use. Also, obviously, the name of the secret agency (“the Super-Secret Central Intelligence Agency”) wasn’t very well considered before it’s placement in the story. These are all issues that need work when modernizing the game.

Figure 4: Rough sketch for the game, digitally drawn using Iskn Slate2 by Ray.

1. Planning for Development of Prototype
2. Basic Plan for Proto type

The Prototype for the new version of the Dinosaur Game (labeled as TNG for The New Game—not to be confused for the television series *Star Trek: The Next Generation*, sorry, nerd joke.) will eventually be developed in the Unity game engine, but will begin as a restoration project of TOG using ,SDL (Simple DirectMedia Layer) 2.0.8, except the title screen will be slightly different. From there, modernized digital artwork by Ray and marias will be used for backgrounds and movable sprite figures (aka assets in Unity, or objects of class movingFigure in the code for the prototype). Further, although the spear will at first generally function as it did in TOG for the initial prototype of TNG, in later stages if prototype development, the spear is to look and fly on the screen much like a real spear would in a parabolic trajectory, rotating through the motions of facing diagonally upward at first when it is initially fired, facing perfectly horizontal at the vertex of the trajectory, and finally facing diagonally downward as it nears its destination.

1. Filling in Pseudocode to Recreate TOG

Recreating and developing TOG is to be as much for the purpose of exercising this Development Team in software development as much as it is for the purpose of creating a game prototype that will eventually become TNG, the modernized version of The Dinosaur Game. With this in mind, there will be numerous versions of TOG coded in various languages and making use of different software engines and libraries and even different compilers for the same language. We can use this as a change to review our skills in coding and developing as well as to see what resources are out there and available for us to use, not to mention which resources work best in specific situations.

To begin, Ray will first take the TOG pseudocode and begin converting it to C++ code for use with the SDL 2.0.8 Library for compilation with Gnu Compiler Collection (GCC). From here, many versions of the game can be made using various languages and environments.

1. Supplemental Paper on Gravitation

In order to facilitate the path of the spear once it is thrown, a software library that simulates the effect of terrestrial gravitation is being prepared along with an instructional supplemental paper, Applying Terrestrial Gravitational Motion in 2D Settings: A Physics Supplement for Application to Software, Including a Primer or Review of Analytical Geometry, Basic Calculus, and Newtonian Kinematics, Plus TerrestrialGravitation, a C++ Library. In this paper is a basic primer and review of some analytical geometry, basic calculus, and basic Newtonian kinematics. The software library gives x and y positions and velocities at each moment as well as implements downward acceleration due to gravity. Also implemented, but not to be used in the game, is bouncing off of the ground due to the Coefficient of Restitution (COR). Initially, TOG will be developed using the original path of the spear—upward at a 45° angle until it gets to the top of the screen, and then straight downward. After a working prototype is created using this path, and the TerrestrialGravitation library is fully tested, then the path of the spear will be changed to one that properly corresponds to one produced naturally by gravity.

1. Conceptual Development of Newer Modernized Game
2. Discussion of Possible Change to Game Title

Ray and marias, in personal conference with each other, have discovered that there are numerous games in existence about dinosaurs in one context or another, with titles all featuring the word “dinosaur.” With this in mind, the thought occurred to Maria that a game with such a generic title as “The Dinosaur Game” may get lost in a Google search or even just a search on Google Play for an Android app, which is one of the forms this game may be developed into. In context of the some of the details of the game story in development (see the next subsection, C. Ideas to Fill in the Story) it was agreed that one possible alternative for a title would be “The Time Traveling Dinosaur Apocalypse.” However, Ray has decided not to finalize this change just yet (by changing the title of the game document, the name the GitHub depository, the names of all the file folders/directories, the filenames themselves, etc. as this would entail some deal of effort and if the name should change again, even more effort would have to be expended, also, keeping all the code and records under different changing names, particularly names of depositories, directories, and files would also cause confusion as to where to find everything. Therefore, while the devilment team will keep in mind that the final name of the game may change, its working title throughout the duration of this project will be “The Dinosaur Game.”

1. Ideas to Fill in the Story

Ray has come up with more ideas to fill in the game story. They are as follows.

1. First Stage/ Volume One

You (aka the player) are a secret agent, with a specific name and agency (real or fictional) to be determined, tasked with an assignment of going back in time to 82,356,743 BC to eliminate intelligent dinosaurs before they travel forward in time and attempt to dominate the modern world. The time travel mechanism used to bring you back in time was developed by these intelligent dinosaurs and was adapted by the agency for use by humans. You will be eliminating the intelligent dinosaurs by throwing SPEARs, or Stick Projectile Eradication and Annihilation Receptacles. Essentially, these are miniature time traveling devices developed by the agency using the technology initially developed by the dinosaurs. They teleport anything they hit, but instead of sending it to a specific time and place, they send it into an abysmal void where nothing exists. Be very careful, these dinosaurs are very intelligent and can outsmart you at any moment. Your first skirmishes will take place on land, and initially your opponents will also be on land, but soon intelligent pterosaurs will join them from the air. Also, initially the terrain will be continuously solid ground, but later on there will be gaps filled with water and lava, as well as hills, mountains, and volcanoes, so watch out. Later battles of this first stage will also be fought in the sea with giant aquatic ancient animals, that are also intelligent, and in the air with more pterosaurs and dinosaurs in giant flying vehicles (the dinosaur equivalent of fighter jets).

Figure 5: Preliminary sketch for the game, digitally drawn using Autodesk Sketchbook by marias.

1. Second Stage/Volume Two

A well-educated, misunderstood and alienated genius from a very rich family who works in animal genetics discovers a giant metallic pod in the woods while hiking one day. S/he (Ray hasn’t determined this person’s gender yet) goes inside and finds a bunch of large foot-petal controls and a three-ring binder with hand-written notes in English telling him/her what the controls do along with suggestions to use this pod device to travel back in time to the time of the dinosaurs to bring back specimens to experiment on. At the end, the notes are signed, “Your Secret Friend.” Also there’s a post-script that reads, “P.S. Whatever you do, hide this time travel pod in your garage and don’t tell *anyone* about it! They won’t understand and you can get into a *lot* of trouble. I know, *I* almost did!”

At first s/he hides the time travel pod in his/her garage, but later s/he builds a secret lab and hides the pod in it. As time goes on, this person carries out many experiments using combinations of dinosaur and modern animal DNA. S/he has a collection of very strange and bizarre animals from monkeys that roar like dinosaurs to tree hopping dogs that can glide through the air. Eventually, s/he comes up with intelligent dinosaurs that can speak.

Some of the dinosaurs tell him/her that they want to leave the lab and go explore, but s/he can’t let them do that because then his/her secret will be out, so as a compromise, they agree to use the pod to go back into the time of the regular (not-so-intelligent) dinosaurs, where they will fit in with their own kind. This person is sad to see his/her friends go, and take the only time traveling device s/he has ever used with them, but knows there is no other solution to this problem. There is also another issue. There is only room enough for one dinosaur, maybe two small ones at the most, in the pod. However, this issue is mysteriously solved as ten more pods spontaneously appear in the lab with a note saying “Compliments of Your Secret Friend!” There is even an extra, mini-pod (human sized, with controls made for human hands) for this person to use after all the dinosaurs leave.

Soon the agency discovers what this person has done and they want to stop him/her from creating the intelligent dinosaurs, so they send a message to the directors of the agency in the past when the person started experimenting and hadn’t made the dinosaurs intelligent yet. You are the main agent sent over to arrest and stop him/her. S/he evades capture and unleashes all his genetic monstrosities to come get you and wreak havoc upon the world. At first, the land animals come for you (with some help from flying ones) and you must send them into oblivion with your SPEARs. Next, this person lets loose all his/her marine abominations into the sea, and you must conquer them as well. Finally, an air battle ensues with the flying beasts again before you must defeat the enemy him-/herself.

1. Third Stage/Volume Three

One day in Pennsylvania 1723, a man discovers a large bag of gold nuggets on his farm along with a note where to mine for more. The great, great grandson of this man, a minor stockbroker in New York 1929, finds a note in his desk telling him to sell all his gold and invest in a series of short positions in stocks just before the stock crash that commenced the Great Depression. Both of these notes were signed, “Your Secret Friend.” The stockbroker ends up making millions of dollars from this trade. With this money, the stockbroker starts his own successful brokerage firm, making even more money. This broker was the grandson of the mad scientist. The agency eventually hears about these incidents and discovers that the parents of the mad scientists are involved as well as some intelligent dinosaurs. You are sent to defeat the dinosaurs, and after them, the parents themselves.

1. Development of the Initial Prototype
2. Initial Stages

Ray first began by turning the pseudocode above into code that can more likely be compiled by the GCC (in this case, using a MinGW [Minimalist GNU for Windows] port of GCC) using the Software Development Kit (SDK) Orwell Dev C++ 5.11. While initially, only vague C++ functions to display the start screen and implement gameplay were the only items planned, Ray decided to create some C++ object classes in order to facilitate more detailed implementation of graphics and functionality. Of primary consideration has been the compatibility of these classes with SDL 2.0 and its various C structures and functions.

The first class to be created was class gameWindow, which facilitates creation, use, and deletion of the main window for the game. It’s variable members (SDL\_Window\* window, SDL\_Surface\* surface, SDL\_Renderer\* renderer, SDL\_Texture\* texture, bool failure, bool \_closed, and bool \_open) are all protected (only accessible by functions in the class itself or by functions from a derived class, if applicable) and are only indirectly accessible by way of using class functions made for that purpose. For example, instead of assigning a value to \*surface directly, one must use the function void Surface(SDL\_Surface\* surf) and insert the desired value in \*surf. Values for members are obtained using applicable functions that require no input parameter. For example, SDL\_Surface\* Surface(void) returns the pointer surface, which can be used to obtain the value of \*surface.

The next class to be implemented was class movingFigure, which was created to facilitate the display, placement, apparent movement, and handling of all the character images in the game. Each object is represented by an array of similar images that, at the appropriate time, are shown in the correct sequence and location in the window of the game and make it appear as though the spear, the dinosaur, and the player are moving throughout the runtime of the game.

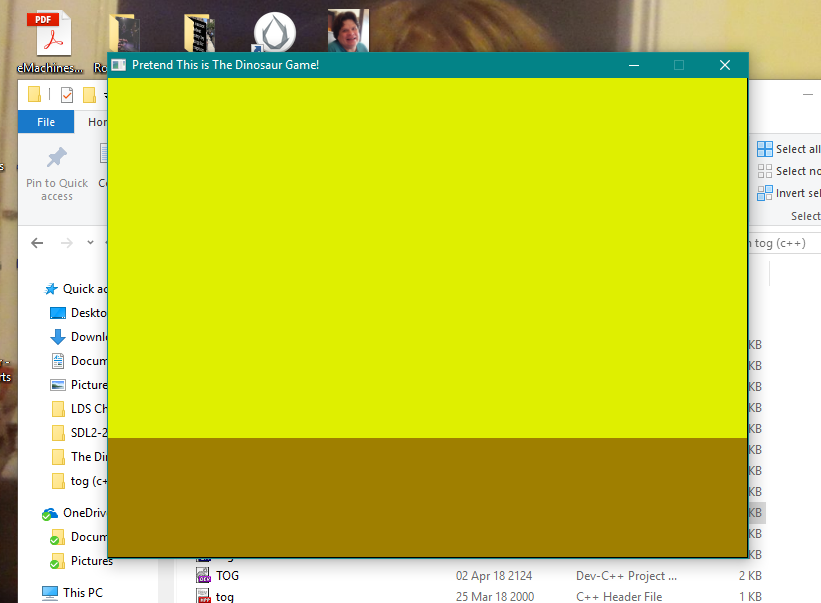
As far as success with compiled and run code, after many frustrating misadventures learning how to appropriately integrate SDL 2 into the IDE package Dev-C++ and how to appropriately include the SDL into the program code, including one four-day long stalemate when nothing would compile because of the use of “SDL\_INIT()” when attempting to call the function “SDL\_Init()”, Ray finally successfully compiled and run his first C++ program using the SDL library rendering the window shown in this section on 3 April 2018. Following this, on 25 June, Ray was able to get a splashscreen showing the logo for The Trashcan Software and Media Publication to display. Subsequently, on 7 July, Ray got a sample start screen (a still screen, unfortunately) for The Dinosaur Game inside a 640- by 420-pixel window to display.

Figure 6: Screen capture showing the result of Ray's first successful compilation of C++ code using the program library SDL 2.0.8.

1. Use of Simple DirectMedia Layer (SDL) and Abandonment of a Class

Due to the implementation of windows, graphics, and controller processing being prohibitively complex to code for without any prepared software library or engine, Ray decided to use the Simple DirectMedia Layer library, version 2.0.8 (or, SDL 2.0.8, or, simply SDL 2) to facilitate these basic functions. Since the beginning of June 2018, Ray has been exploring the use of the sublibrary SDL Image 2 (basically, a library made for use with the SDL 2 library to augment its functionality with graphics and images, particularly any of those not stored in a Windows Bitmap (BMP) format. However, it may not be necessary to make use of this sublibrary as all image files for the game are either already in BMP format, or can easily be imported to BMP format.

Another issue arose when, since early July 2018, Ray has been attempting to devise a way to display text along with graphics on the game window. Initially, he conceived of a class he called class RasterText (a raster is a rectangular pattern for displaying pixels, hence, this class got its name because Ray planned to implement the display of text characters using raster patterns), which was going to take a pre-typewritten file saved from Windows Paint that contained all the printable ASCII characters (beginning at 0x20 or character 32, the space) going all the way to the end of the set, split this long rectangle into up to 128 shorter rectangles, save them into an array of character images—of which there would be another array of 8 for each different possible font, resulting in a 2-dimensional array of 8 by 128. Each one of these character images would be available for recall at any time to be placed anywhere on the game window.

This would have been quite an implementation. Nevertheless, once, on 10 July, Ray discovered another sublibrary, known as SDL TTF 2 (SDL’s sublibrary for True Type Font rendering, or just TTF), he decided to abandon class RasterText in favor of using the TTF sublibrary. The implementation of the class would entail creating 8 files in different fonts that would all have rectangles of characters that can be made as long as possible, having the code split them all up into large arrays, consuming quite a bit of memory, and then having to convert desired characters into integers in order to facilitate addressing the arrays so that the correct characters can be located in them so that they can be correctly displayed in the game window. However, implementing the TTF sublibrary and using it to place text in the game window would only entail the creation of a single much smaller rectangle that would represent the area in which the text is to be placed and the use of a short character string (the desired text) in the correct place in the code. Therefore, class RasterText shall be abandoned. In its place will most likely be a function in one of the non-class files that will place text in the game window making use of SDL TTF 2.

1. Sprite Image Assets

On 16 April 2018, Ray began creating sprite images for the TOG prototype using a combination of Windows Paint, Microsoft Office Picture Manager (MOPM), and the GNU Image Manipulation Program (GIMP) 2.8.22. A considerable number of images are necessary for this game to present the illusion of 2D animation while the dinosaur is walking toward the player, the player is standing still and walking toward and away from the dinosaur (with and without spear in hand), and, perhaps most importantly, when the spear is thrown and in motion across the screen in the hope of it colliding with the dinosaur.

While the prototype will only make use of 315° (-45°) to 45° (from the horizontal facing right) images of the spear, images are being made of the spear for every 15° interval from 0° (angle 000) to 345° (angle 345, 15° away from the full 360° circle of rotation) in preparation for use in the final version. As of 18 April, 24-bit color Windows Bitmap (BMP) images are being prepared using a combination of Windows Paint and GIMP and then the images are being flipped horizontally and vertically also using both Paint and GIMP. Rotation of the spear images by angles that cannot be obtained by mere horizontal and vertical flipping is being done using a combination of MOPM and GIMP. Conversion of the 24-bit color images to 32-bit color images that include transparency is done using GIMP. Since GIMP seems to be the most capable program to handle everything that needs to be done to the image assets from creation to flipping to rotation to conversion to include transparency, after Ray becomes proficient using GIMP, he will probably continue to use it for most, or even all, image creation and manipulation from this point onward and may abandon use of Paint and MOPM altogether.

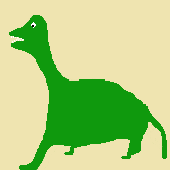
Because the 32-bit color BMP format includes 256 shades of transparency (8 bits, plus 256 shades of red, green, and blue, or 8 bits for each color, for a total of 24 bits before the transparency and 32 after) and the needs of the game are somewhat limited on color as well as transparency—256 shades of transparency are not needed, but simply one bit to determine whether a pixel is completely transparent or completely opaque, Ray may opt to export all the image sprite assets to 16-bit color BMP, with 5 bits used per color (R-G-B, for a total of 15 bits) plus one bit to determine complete transparency or complete opacity of each pixel. This can help conserve space and make for a more compact total file size for the completed prototype as well as for the final version of the game.

Figure 7: A sample BMP images of the player, the dinosaur, and the spear, all digitally drawn by Ray using Windows Paint.

On 19 April, Ray created an animation of the spear flying through the air using Pencil2D 0.6.1.1 and released it as a moving GIF on Facebook. On 24 June, Ray created an animation of the prototypical dinosaur walking using Pencil2D and released it as a moving GIF on Facebook.

# Appendix 1: Letters for Designation of Stage of Software Development Cycle

The terms “alpha” and “beta,” as it applies to software development, were first used by Martin Belsky of IBM in the 1950s. Some software publishers and developers over the years have made use of “gamma” and “delta” to designate later stages of software development and testing. Ray Arias invented the use and placement of all the other Greek letters, as well as all of the designations below.

ν Nu (v): In the Planning (noein) Stages of Development (Minimal or No Working Code)

π Pi (n): In First (protos) Stages of Code Development

ω Omega (w): Raw (omos) Program/Secondary Development

α Alpha (a): Unstable (astathos) Program/Primary Testing

β Beta (B): Secondary Testing (Some Major Features May Not Work)

γ Gamma (y): Tertiary Testing/Stable Except Appearance and/or Minor Features

δ Delta (d): Stable Except Minor Features

ε Epsilon (e): Fully Stable (eustathos) and Fully Tested (the only allowable bugs are for unlikely situation and very minor defects)

υ Upsilon (u): Software Version That Has Been Left Behind (hypoloipos) and Is Now Obsolete

# Appendix 2: Development Log

Date Name Development Work

9 February 2018 Ray Game Document commenced

12 February 2018 Ray Section I. History written

15 February 2018 Ray Google Docs folder created and shared with Stan, files initially uploaded

20 February 2018 Ray GitHub repository opened and files initially uploaded

20 February 2018 Ray Game Document copied to GitHub Wiki

23 February 2018 Ray Section II. Planning for Development of Prototype commenced

24 February 2018 Ray Subsection II A. Basic Plan for Prototype written

24 February 2018 Ray Subsection II B. Discussion of Possible Change to Game Title written

24 February 2018 Ray Subsection II C. Ideas to Fill in the Story commenced

26 February 2018 Ray Began opening issues in GitHub to begin organizing work on game

26 February 2018 Ray Appendix 1: Development Log commenced

26 February 2018 Ray Subsubsection II C 3. Third Stage/Volume Three written

1 March 2018 Ray Digitally drew rough sketch for game and placed into Document

2 March 2018 Ray Drawing by marias copied into project files and placed into Document

2 March 2018 Ray Updated issues and Wiki on GitHub

8 March 2018 marias Created drawing of giganotosaurus

9 March 2018 marias Created drawing of Apatosaurus

10 March 2018 Ray Began writing supplemental paper on how to apply terrestrial gravitational physics to software and games.

16 March 2018 Ray Inserted marias’ drawings into document

16 March 2018 Ray Inserted new logo for The Trashcan and cover art for Game Document and Development Log

17 March 2018 Ray Showcased cover art on black page (black rectangle in background made the size of the page)

24 March 2018 Ray Edited pseudocode file saved as TOG.pseudo and converted code to conform with C++ standards and saved as TOG.cpp

25 March 2018 Ray Researched Windows- and Android-compatible graphics libraries and chose SDL 2.0 for encoding and testing TOG.

26 March 2018 Ray Broke Section II. Planning for Development of Prototype Into two sections consisting of above and Section III. Conceptual Development of Newer Modernized Game.

3 April 2018 Ray First successful program compiled and run giving window rendering TOG background. (Insert your favorite version of Leonard Cohen’s “Hallelujah” here; Praise Satan!)

14-16 April 2018 Ray Rewrote original game music in GNU LilyPond 2.18.2.1 to sound more like original music and inserted sheet music and code for sheet music into Document.

16 April 2018 Ray Began creating 100 X 100 pixel sprites (graphic figures) for player in 24-bit color and 256 color BMP formats using Windows Paint. Figures made for player standing still, walking left, and walking right both with and without a spear.

17 April 2018 Ray Continued creating sprite assets in 24-bit color and 32-bit color with transparency using Windows Paint, Microsoft Office Picture Manager, and GIMP 2.8.22. Figures made for spear at angles (in degrees from horizontal facing right) 000, 015, 045, 075, 090, 105, 135, 165, 180, 195, 225, 255, 270, 285, 315, and 345. Transparency added to some of player images as well as spear images. All images made or transformed in GIMP were saved in XCF format as well as exported to 24-bit color BMP format as well as 32-bit color BMP format with transparency.

17 April 2018 Ray Began writing subsection C. Sprite Image Assets in section II. in order to describe the process of sprite image creation, manipulation, and conversion to include transparency as well as their use in the TOG prototype and possible use in the final modernized TNG final version of the game.

18 April 2018 Ray Completed creating images of spear at every 15-deg rotation for every angle around the circle, for a total of 24 images of spear per format. At this point, there is a complete set of images for 24-bit color BMP, and GIMP XCF format. Formats yet to be done: 32-bit color BMP including transparency, 256-color BMP, and 16-bit color BMP including 1-bit transparency. Images of player standing still, walking left, walking right, all with and without spear are already done in all these formats, except 16-bit color BMP including 1-bit transparency. Made comments on file formats and file size of image sprites in subsection C. of section II.

19 April 2018 Ray Created animation video in Pencil2D of spear flying in air and released it in Facebook as post in Stan’s account.

29 April 2018 Ray Completed writing main part of supplemental paper (2D Terrestrial (Nonturbulent) Gravitational Motion) on how to apply terrestrial gravitational physics to software and games. Ray decided to also begin a C++ software library for applying these principles. Supplemental paper and accompanying library are planned to become part of this game.

29 April–20 May 2018 Ray Expanded supplemental paper by attaching Library Guide as appendix, added tutorial on calculus and expanded tutorial on gravitational physics. Retitled paper as Applying Terrestrial Gravitational Motion in 2D Settings. Also, continued to design and code accompanying C++ library TerrestrialGravitation.

29 May 2018 Ray Added subsection D. to section II. regarding the supplemental paper and accompanying software library. Worked on routines inside of class tog.

1–3 June 2018 Ray Added class movingFigure to game code, for animated sprites, to facilitate in positioning and animating sprites.

23-24 June 2018 Ray Took a primitive 2D drawing of a green dinosaur and turned into a set of 3 drawings for animation. I put them into a .gif file and put it out to Facebook.

25 June 2018 Ray Successfully compiled code that displays a splashscreen showing the logo of The Trashcan Software and Media Publication.

7 July 2018 Ray Successfully compiled code that displays the title screen for The Dinosaur Game.

12 July 2018 Ray Inserted new section, Section IV. Development of Initial Prototype, as well as Subsection A. Initial Stages.

12 July 2018 Ray Moved Section II, Subsection C. Sprite Image Assets to Section IV, Subsection B. This makes Section II, Subsection D. Supplemental Paper on Gravitation the new Subsection C.

12 July 2018 Ray Inserted Letters for Designation for Stage of Software Development Cycle as Appendix 1. This makes this, the Development Log Appendix 2 now, instead of Appendix 1 (and the only appendix) as it used to be.

13 July 2018 Ray Inserted new subsection in Section IV. Development of Initial Prototype. Subsection B. Use of Simple DirectMedia Layer (SDL) and Abandonment of a Class is the new subsection. This sections describes Ray’s recent attempts to create a class for displaying characters in the game window before later abandoning this class in favor of using the SDL TTF 2 sublibrary instead. The former Subsection B. Sprite Image Assets is now Subsection C.

13 July 2018 Ray Inserted samples of sprite images for player, dinosaur, and spear in Section IV, Subsection C. Sprite Image Assets.

20 July 2018 Ray Placed this project and the TerrestrialGravitation project on hiatus due to overriding household concerns.

28 November 2018 everyone Moved from old venue in Urbana to new venue in Chicago.

6 February 2019 Ray Reactivated TerrestrialGravitation project in new venue.

11 February 2019 Ray Moved older version of Supplemental Paper into newly created subdirectory Old Docs and current version of Supplemental Paper and all the pictures that go with it into newly created subdirectory Current Doc.

12 February 2019 Ray Reactivated this project in new venue. Moved older version of this document into newly created directory Old Docs and current version of this subdocument into newly created subdirectory Current Doc.

15 February 2019 Ray Inserted and wrote Abstract into Supplemental Paper.

15 February 2019 marias Digitally drew cover picture for Supplemental Paper.

16 February 2019 Ray Inserted cover picture into cover page of Supplemental Paper, labeled main sections, and prepared Table of Contents for Supplemental Paper, renumbered pages in this document (first nine pages are in miniscule Roman numerals and after that each page is Arabic numerals, beginning with Page 1), and inserted space for an Abstract.

17 February 2019 Ray Retitled Game Development Guide at beginning of this document (Section 4O) to Guide to Terminology Specific to This Document, which the first part (now subsection A) is Development Team, and inserted a second part (subsection B), Planned Versions and Supplements.

17 February 2019 Ray Wrote Abstract, inserted blank pages between cover page (page i) and first title page, between first title page and second title page, between second title page and copyright notice page, between copyright notice page and Guide to Terminology Specific to This Document (Section 4O), between Section 4O and Table of Contents (Section OOO), between Section OOO and Table of Illustrations (Section OO), and between Section OO and Abstract (Section O on page 1), and reindexed Section OOO and Section OO.